Linear Time Sorting II

Radix Sort, Bucket Sort
Radix Sort

- Decimal numbers – 10 unique digits
- Any $d$-digit number takes $d$ columns
- Key concept – Sort input by least significant digit first, then by the second least significant digit and so on until we are done sorting on all $d$ digits.
- At this point all numbers are sorted
- $d$ passes
<table>
<thead>
<tr>
<th>329</th>
<th>720</th>
<th>720</th>
<th>329</th>
</tr>
</thead>
<tbody>
<tr>
<td>457</td>
<td>355</td>
<td>329</td>
<td>355</td>
</tr>
<tr>
<td>657</td>
<td>436</td>
<td>436</td>
<td>436</td>
</tr>
<tr>
<td>839</td>
<td>457</td>
<td>839</td>
<td>457</td>
</tr>
<tr>
<td>436</td>
<td>657</td>
<td>355</td>
<td>657</td>
</tr>
<tr>
<td>720</td>
<td>329</td>
<td>457</td>
<td>720</td>
</tr>
<tr>
<td>355</td>
<td>839</td>
<td>657</td>
<td>839</td>
</tr>
</tbody>
</table>

**Figure 8.3** The operation of radix sort on a list of seven 3-digit numbers. The leftmost column is the input. The remaining columns show the list after successive sorts on increasingly significant digit positions. Shading indicates the digit position sorted on to produce each list from the previous one.
\textbf{Radix-Sort}(A, d)

1. \textbf{for} $i \leftarrow 1$ \textbf{to} $d$
2. \textbf{do} use a stable sort to sort array $A$ on digit $i$
Key points

• The sorting algorithm used for each digit must be stable

• Otherwise we risk shuffling the sorted order of the previous digit

• Any good candidates??
Complexity

**Lemma** Given $n$ $d$-digit numbers in which each digit can take on up to $k$ possible values, RADIX-SORT correctly sorts these numbers in $\Theta(d(n+k))$ time
Bucket Sort

• Linear time for an input drawn from a uniform distribution
• Assume input is distributed in the interval \([0,1)\)
• *Key concept* – Divide the interval into \(n\) equal sized buckets and distribute the input in them
• We can then sort each bucket
**Bucket-Sort(A)**

1. \( n \leftarrow \text{length}[A] \)
2. \( \text{for } i \leftarrow 1 \text{ to } n \)
   - \( \text{do insert } A[i] \text{ into list } B[\lfloor nA[i] \rfloor] \)
3. \( \text{for } i \leftarrow 0 \text{ to } n - 1 \)
4. \( \text{do sort list } B[i] \text{ with insertion sort} \)
5. \( \text{concatenate the lists } B[0], B[1], \ldots, B[n - 1] \text{ together in order} \)
Figure 8.4 The operation of BUCKET-SORT. (a) The input array $A[1..10]$. (b) The array $B[0..9]$ of sorted lists (buckets) after line 5 of the algorithm. Bucket $i$ holds values in the half-open interval $[i/10, (i + 1)/10)$. The sorted output consists of a concatenation in order of the lists $B[0], B[1], \ldots, B[9]$. 
Complexity

- Except line 5 everything takes $O(n)$
- So how long does line 5 take?